ESTRUCTURAS DE DATOS

Curso 2018/19

PRÁCTICA 10

Instrucciones

- *Se debe completar en una sesión.*
- *Práctica individual.*
- *Lee el enunciado completo antes de comenzar. Los comentarios incluidos en el código también pueden proporcionar información útil.*
- *Se habilitará una tarea para que entregues el código desarrollado.*
- *La práctica será APTA si se superan todos los test de validación proporcionados.*

Grafos

Vamos a utilizar grafos no dirigidos para resolver un laberinto. Los grafos los vamos a representar utilizando matrices de adyacencia **`EDMatrixGraph.java`** y listas de adyacencia **`EDListGraph.java`**. El fichero **`EDGraph.java`** tiene el interfaz con las operaciones del grafo.

Ejercicio 1

Añade un nuevo constructor público a la clase **`EDListGraph`** que dado un grafo de tipo **`EDMatrixGraph`** lo construya utilizando listas de adyacencia. La clase **`EDMatrixGraph`** tiene un método `HashMap<T,Integer> getNodes()` que devuelve un mapa con los nodos del grafo y sus posiciones en el vector de nodos y un método `boolean[][] getAdjacencyMatrix()` que devuelve la matriz de adyacencia.

Ejercicio 2

Implementa un método público en la clase **`EDListGraph<T>`**, **`Set<T> degree1()`**, que devuelva el conjunto de nodos del grafo cuyo grado de entrada es 1.

Ejercicio 3

Implementa el mismo método público que en el ejercicio 2 pero para la clase **`EDMatrixGraph`**, **`Set<T> degree1()`.**

Ejercicio 4

Se va a utilizar el grafo para representar y resolver un laberinto como el de la figura 1. Su representación como grafo sería la figura 2.

Añade un método público a la clase **`EDListGraph`** que devuelva la lista de nodos en orden que forman un camino de salida del laberinto dado el nodo de entrada (`entry`) y el nodo de salida (`exit`): **`List<T> findPath(int entry, int exit)`**. Recordar que los nodos `entry` y `exit` son identificadores enteros de los nodos. El algoritmo **`findPath`** es una modificación del algoritmo de búsqueda en profundidad (**`dfs(int start)`**), puesto que cuando encuentre el nodo de salida no es necesario continuar. Además, a partir del vector de caminos que devuelve el algoritmo de recorrido, construirá la lista que hay que devolver.

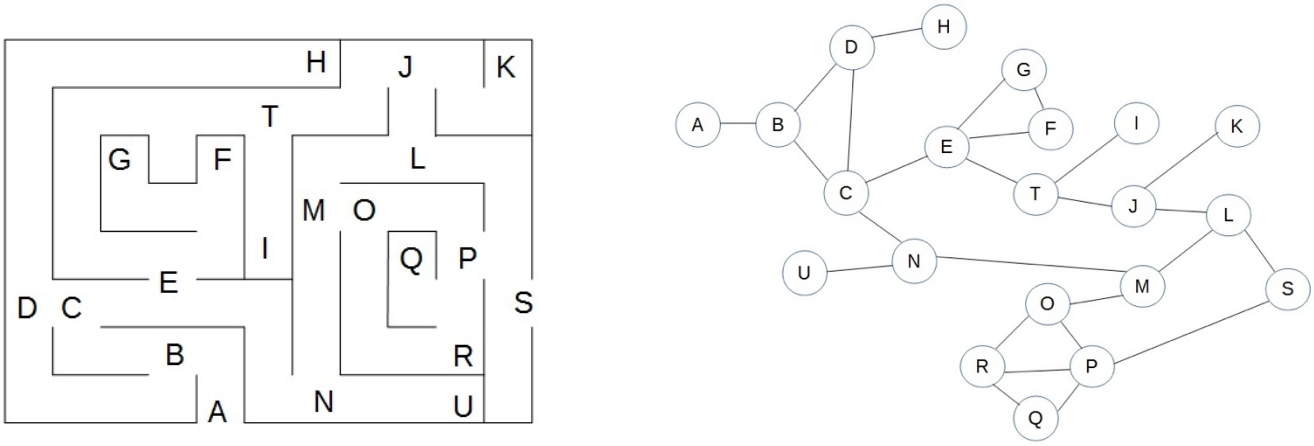
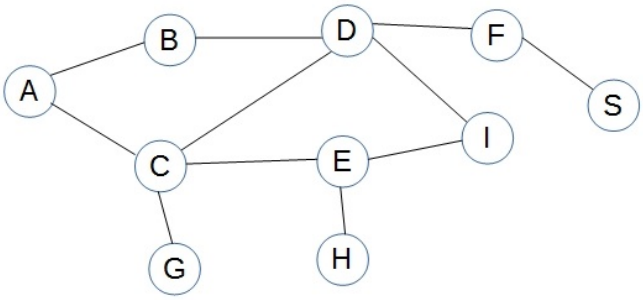
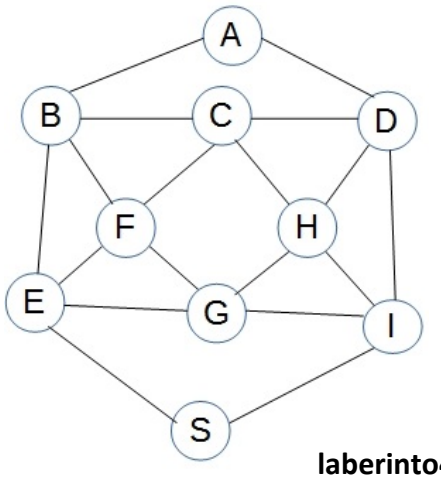


Figura 1: Un laberinto y el grafo que lo representa. La entrada está en el nodo A y la salida en el nodo S.

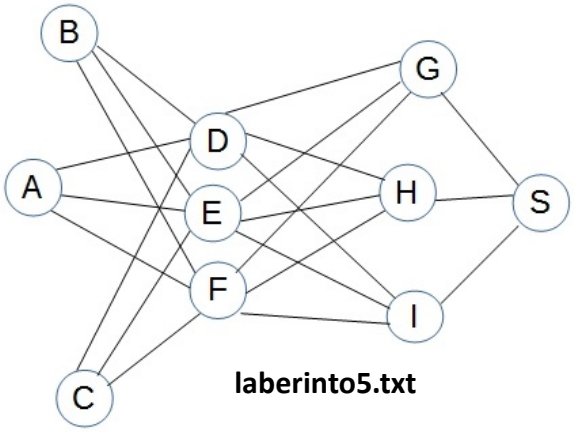
El test usa una serie de ficheros que representan los siguientes grafos:



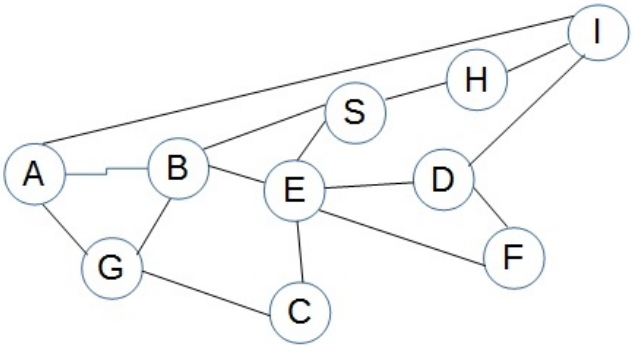
laberinto2.txt



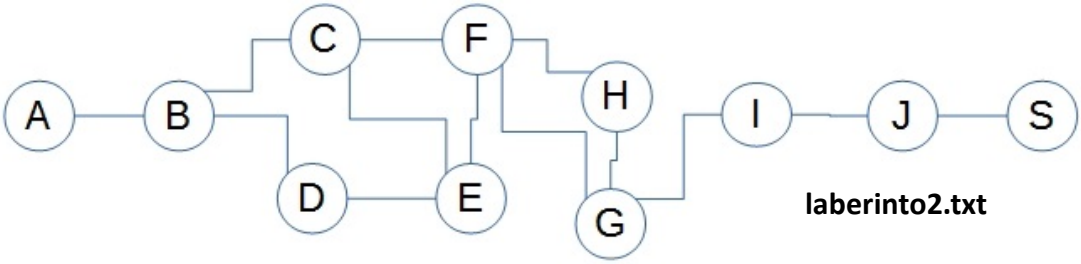
laberinto4.txt



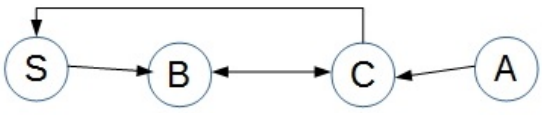
laberinto5.txt



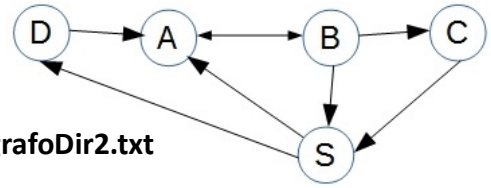
laberinto6.txt



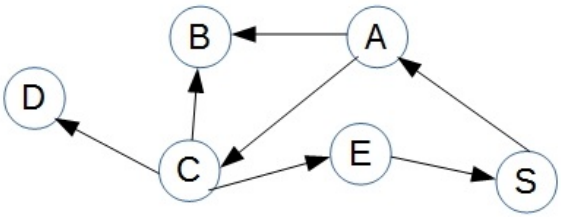
laberinto2.txt



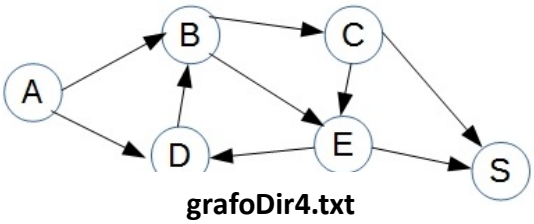
grafoDir1.txt



grafoDir2.txt



grafoDir3.txt



grafoDir4.txt